The Configurable Processor Company

High-Performance Multicore Video Decoder Technology Preview

Gülbin Ezer

Hardware Design Manager

What is the Tensilica Video Engine?

- **2-core SD resolution video decoder IP package (h/w and s/w)**
  - 720 x 480 @30 fps (NTSC)
  - 720 x 576 @25 fps (PAL)
- **Target applications are SDTV, digital camcorders, PVRs, personal media players, satellite and IP TV**
- **Programmable and multiformat**
  - H.264 main profile @ L3 decoder
  - MPEG-4 ASP [no GMC] @ L5 decoder
  - MPEG-2 MP @ ML decoder
  - VC-1 (WMV9) main profile decoder
- **Joint development with Ittiam Systems**
  - Tensilica designs programmable h/w and instruction extensions
  - Ittiam designs s/w codecs

Design Goals

- **Technology: 0.13µ G process, < 200MHz**
- **Area: < 10.5mm^2**
- **Easy system integration**
- **Reduced requirements from system memory controller**
  - < 50% 16-bit DDR @200MHz
- **Multiformat support (h/w-supported acceleration)**
  - Cabac
  - Cavlc
  - Deblocking
  - Transforms
  - Motion compensation

Tradeoffs Considered

How many cores?

- Profile the application on the Xtensa cycle-accurate ISS model for a variety of streams (bitrates, complexity)
- **=> Two cores for low clock speed/power/process**

Homogeneous or heterogeneous?

- Homogeneous configuration: duplication of computational resources, more h/w cost
- Heterogeneous configuration: fits algorithm partitioning neatly, less h/w cost
- **=> Dual heterogeneous cores with intelligent DMA engine**

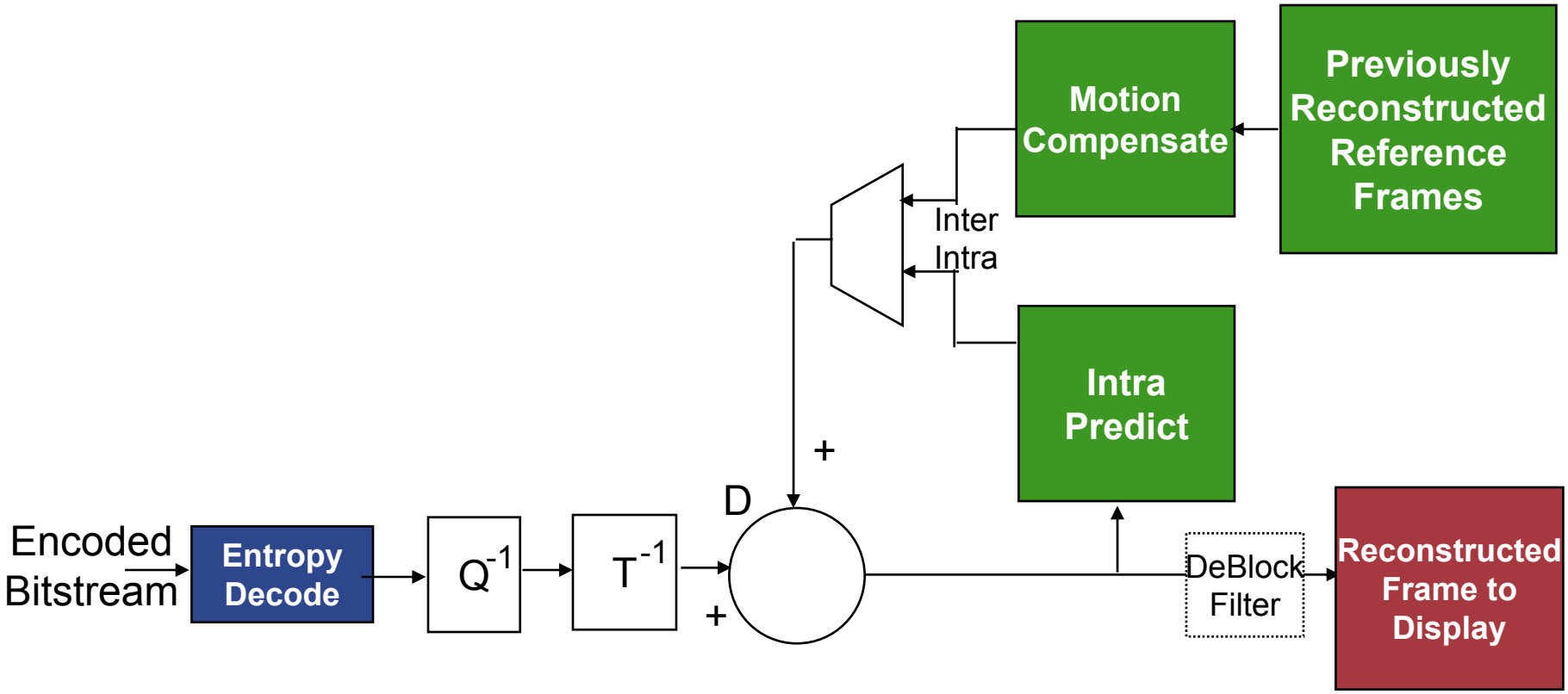
Tradeoffs Considered

- **Hard-wired RTL accelerators or TIE (Tensilica Instruction Extensions)?**
  - Hard-wired RTL: State machine is not a reusable resource, communication overhead, schedule risk, bugs can't be fixed post-silicon
  - TIE instructions: both instructions and state are reusable, no communication overhead, bugs may be fixed post-silicon
  - **=> TIE instructions and state**

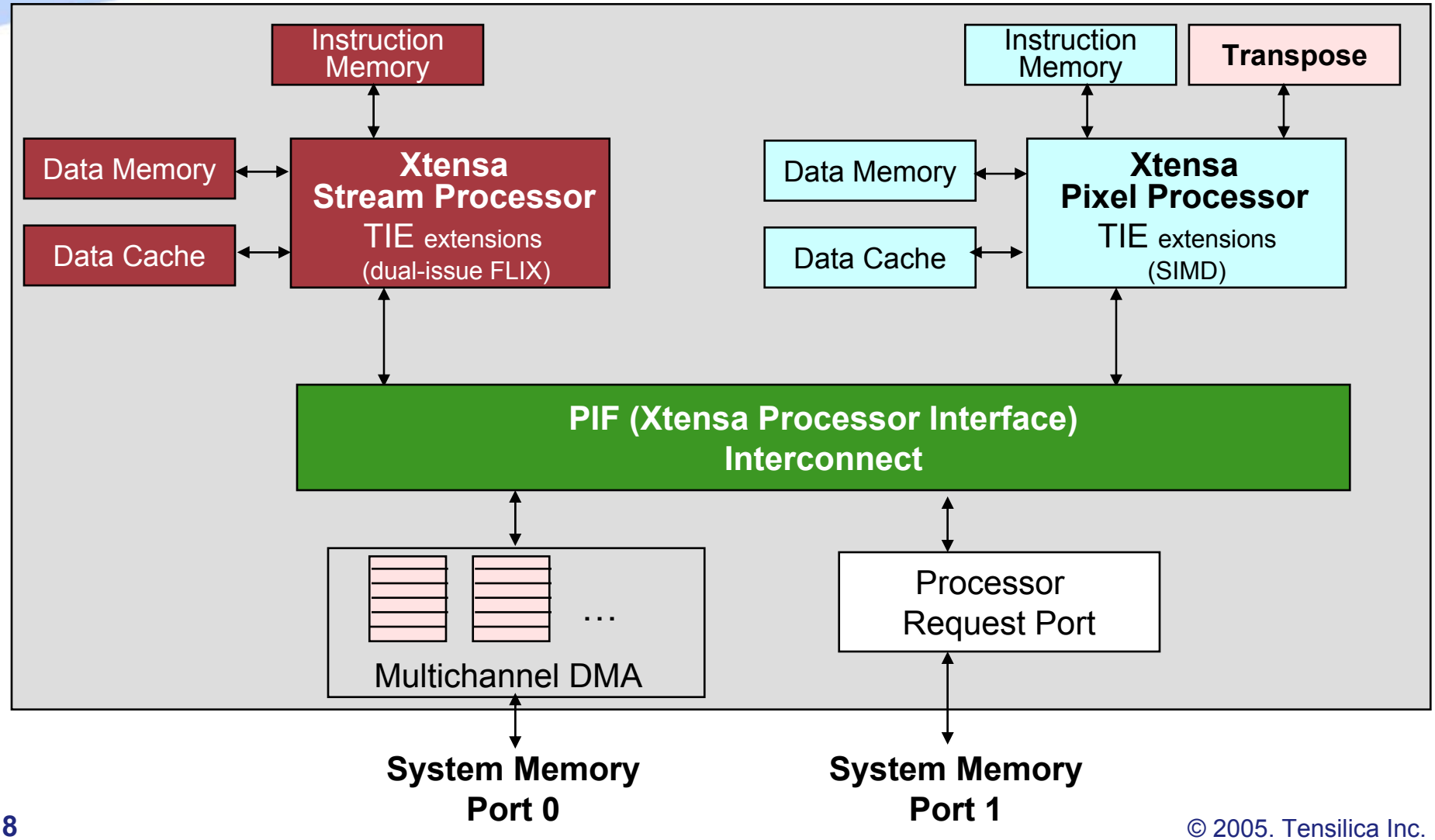
Video Codec Engine Architecture Summary

- **Based on Xtensa LX architecture:**
  - Flexible instruction widths
  - Custom I/O ports
- **State machines implemented using custom instructions**
  - Bugs can be fixed post-silicon, in software
- **Accelerator RTL automatically generated from high-level instruction descriptions**
  - Significantly reduces design/verification schedule
- **Custom instructions reusable by multiple codecs**
- **Intelligent DMA engine hides memory latency**

Video Decoder Data-Flow Path



Top-Level Block Diagram



Description of Stream Processor

- **Functions**
  - Accelerated for serial processing (entropy decoding, motion vector prediction, etc.)
  - Set up reference DMA using decoded motion vectors
- **Stream Processor configuration**
  - 32K data memory, 4K data cache, 30K instruction memory
  - 64b instruction width (16/24/64b instructions), 64b data memory/PIF width
  - DMA path to both data/instruction memories
- **Stream Processor TIE extensions**
  - FLIX for dual-issue of instructions
  - TIE ports to system interface (interrupt/status output)
  - TIE instructions for function acceleration

Description of Pixel Processor

Functions

- Minimal control code
- Accelerated for pixel-wise processing using SIMD
- DMA-reconstructed frames to external memory

Pixel Processor configuration

- 64K data memory, 4K data cache, 20K instruction memory
- 24b instruction width (16/24b instructions), 128b data memory/PIF width
- DMA path to both data/instruction memories

Pixel Processor TIE extensions

- SIMD instructions for function acceleration
- TIE queue to transpose pixel data
- TIE ports to system interface (interrupt/status output)

Instruction Extensions

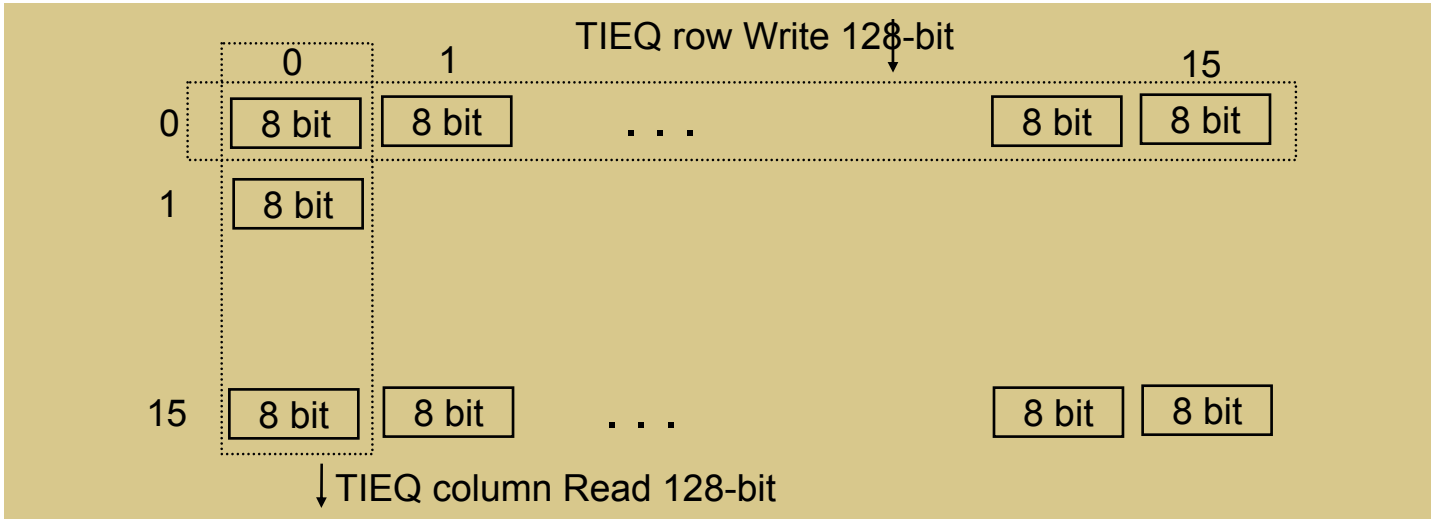
Operations	Slot	Description
cmpgt_i, cmpge_i, cmplt_i, cmple_i, cmpeq_i, cmpgtt_i, cmpget_i, cmpltt_i, cmpltt_i, cmpeqt_i, cmpeqt_i, cmpgtf_i, cmpgef_i, cmpltf_i, cmplef_i, cmpeqf_i, mvt, mvf	0	Conditional compare/move
wrnibble, rdnibble, rdbyte	0	Misc load/store
ldz	0	Misc
bs_loadbuffer, bs_getbits, bs_ldgetbits, bs_showbits, bs_flushbits, bs_AlignByte, bs_flushbitsstate, bs_ldz	0	Bitstream load/store/merge/align
bs_runbefore, bs_invscan, bs_ldtable8, ld_quant, bs_rdtones, bs_quant, bs_suffixsixe, bs_levcode, bs_calclevel, bs_calcindex	0	CAVLC
init_cabac, ld64_cntxt0_i, st64_cntxt0_i, ld64_contxt1_i, st64_cntxt1_i, ld64_cntxt2_i, st64_cntxt2_i, bs_rdlevel, st64_level_i	0	CABAC
bs_decisionbin, bs_bypassbin, bs_readcoeff4x4	1	CABAC
ldMVleft, ldMVtop, lvMVtopright, ldLeftRefFrame, ldTopRefFrame, ldTopRightRefFrame, mv_predict, getCurMVptr, getLeftMVPtr, getTopMVPtr, getTopRightMVPtr, getNeighborIndex	0	MV Prediction
init_leftnnz, check_cbp_bit, calc_cntxtincr, modify_csbpnnz, read_csbp_bit, calcN	0	Coded block pattern
mbBsInit, mbBsLoad, mbBsGetblkIndx, mbBsMvdiff, mbBsRefdiff, MbBsCalc	0	Boundary strength
ldz, calcMvXY, calcFrmXY, mvXYClip3, MvDLoad32, DMvClip3, DMvPow2Div, DMvAbs16, DMvCompUci, MvMacLLHS, MvMacLLSScale6, MvMacHLSScale8, MvMacLLSScale8	0	Misc MAC/ALU/Clip/Scale
ABS, ADD, ADD.N, ADDI, ADDI.N, ADDMI, ADDX2, ADDX4, ADDX8, NEG, SUB, SUBX2, SUBX4, SUBX8, AND, OR, XOR, SLL, SLLI, SRA, SRAI, SRC, SRL, SRLI, SSA8B, SSA8L, SSAI, SSL, MOV.N, MOVEQZ, MOVF, MOVGEZ, MOVI, MOVI.N, MOVLTZ, MOVNEZ, MOVT, EXTUI	1	Duplicated core ALU for dual issue

Instruction Extensions (Cont.)

Operations	Slot	Description
tez4x8to4x4, index4x8, alignaddr	0	Misc
ld_sr192h_i, st_sr192h_i, mv_sr192	0	SIMD regfile fill/spill
ld_128u_i, st_128_i {_x,_xu,_iu}, mve_24to8x24_i	0	SIMD 128b load/store
ld_64u_i, st_64_i, ld_m8x8u_i, st_m8x8_i {_x,_xu,_iu}, lda_m8x8u_i {_x,_xu,_iu}, lda_m8x8u_i {_x,_xu,_iu},	0	SIMD 64b load/store
ld_m4x8u_i, st_m4x8_i {_x,_xu,_iu}	0	SIMD 32b load/store
mve_24to8x24_i, mv_4x8to8x24u, mve4e_8x24to8x24u, int_2x8x24to16x12, deint_16x12to8x24_i, mvf_16X12, mvt_16X12, duplex16_12to16x12, duplex8_24to8x24, duplex4_4x8to16x12u, conv4e_8x24to4x8_i, sext_8x24,	0	SIMD RF – AR RF move/convert/duplicate
add_16X12 {_i}, sub_16x12, neg_16x12, sll_16x12 {_i}, sra_16x12 {_i}, tge_16x12 {toAR}, tlt_16x12 {toAR}, mv_sr16x12tomax, mv_sr16x12tomin, sat_16x12, sat8u_16x12, subabs_16x12, add_8x24 {_i}, addx4_8x24, sub_8x24, neg_8x24, sll_8x24 {_i}, sra_8x24 {_i}, wr_vesar {_i}, rd_vesar, setshift64, sverc_8x24 {_u}, sat8u_8x24, seli_8x24, sel_8x24, mac_8x24_ar {_u}, mac_8x24 {_u}	0	SIMD ALU/MAC
RdTQ128u, WrTQ128, WrTQReset, WrTQWrAddr, WrTQRdAddr	0	TIE Queue Transpose control

Description of Other Blocks

- **TIE queue transpose block**
  - Attached to Xtensa TIE queue port and controlled by TIE instructions
  - Transpose one 16x16 8-bit matrix (or 8x8 matrices of 16-bit data) in a single cycle



Description of Other Blocks

DMA Controller

- Five channels, chaining DMA, continuous DMA with circular descriptor buffer control
- Arbitration scheme: software-controlled per channel bandwidth allocation priority

PIF interconnect

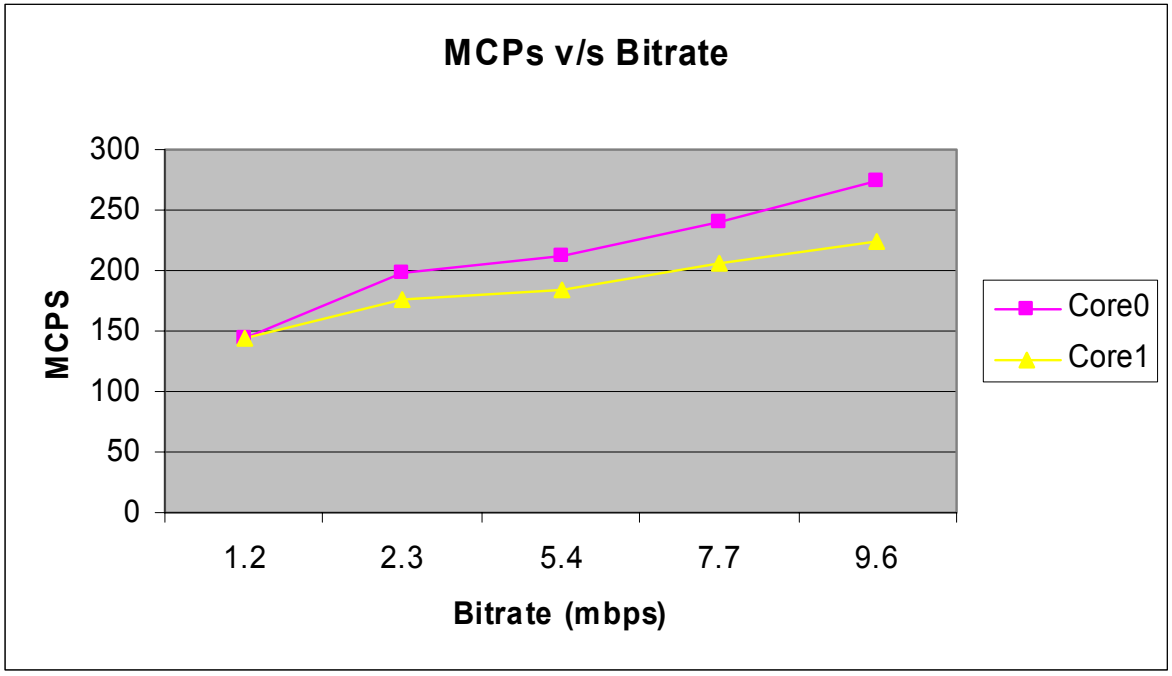
- PIF: full duplex, high-bandwidth Xtensa interface protocol
- Interconnect supports simultaneous transfers between cores, DMA, and memory ports, (3.2GB/s peak bandwidth)

TIE Acceleration Example: CABAC

- **Context-adaptive binary arithmetic coding (CABAC) achieves higher compression in H.264**
- **CABAC requires a large number of cycles on a RISC processor (800Mcycles for a D1 stream)**
- **Others implement CABAC using:**
  - Gigahertz general-purpose processor
  - External hardware accelerator
- **Our solution: TIE-based hardware accelerator using two-slot 64-bit instructions**
- **Peak decode performance is one bin per cycle**
- **CABAC decoding of one 4x4 transform block:**
  - Original C code: **710Mcycles/sec** (requires 710MHz core)
  - TIE accelerated: **13Mcycles/sec** (requires only 13MHz)
- **Area cost: 27K gates (20K estimate after optimization)**

H.264 Profiles for Different Bitrate Streams

- **H.264 decoder profiled for different bitstreams**
  - Most complex standard, implemented first
  - Profiled on cycle-accurate simulation model for the processors
  - Stream Proc (Core0), Pixel Proc (Core1)



TIE Extension Results: Profile Comparison

	Baseline Xtensa Performance (without TIE Acceleration)		Performance on Optimized Xtensa Cores (with TIE)		Processor Size
	Rugby Cabac 5.4Mbps	Rugby Cavlc 5.4Mbps	Rugby Cabac 5.4Mbps	Rugby Cavlc 5.4Mbps	Gate Count
Stream Processor	1.3 GHz	701 MHz	196 MHz	188 MHz	137,840 gates
Pixel Processor	1.22 GHz	1.22 GHz	187 MHz	185 MHz	148,711 gates

Without TIE acceleration, solution would require dual 1.3GHz cores

With TIE acceleration (custom instructions), solution needs only dual 200MHz cores – saves significant area & power

Summary: Why a Pure Software-Based SD Multiformat Video Decoder?

- **Time to market**
  - Pre-implemented, preverified hardware/software solution speeds up chip design
  - Full software programmability allows faster tapeout (fully programmable post-silicon)
- **Flexibility**
  - Full software programmability allows easy adaptation to rapidly evolving video standards and solutions
- **Low cost**
  - Hits performance goals in mature, low-cost 130nm G process
  - Small die footprint
- **Low power**
  - Compact design, low clock rate (< 200MHz), low power with WaitMode

Future Plans

- Encoder package is also coming
- Four heterogeneous processor core solution
- Supports all leading encoders via software
- Simultaneous, multistream encode/decode including transcoding (with both packages)